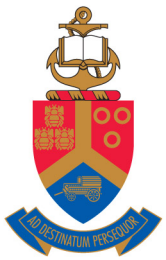

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

COS314 Assignment 2 Report

Giovanni Joubert (u18009035)

14 May 2020

1 Java Application implementing GA on Sudoku

1.1 Representation Used

Each Chromosome in the population (attempt at a Sudoku problem) is represented as 9 char arrays of length 9, creating the rows of the Sudoku board.

1.2 Initial Population Generation

Valid permutations of 9 are inserted into each row (keeping the provided input fields fixed). The permutation is randomized to ensure a diverse population. Keeping the rows intact ensures that there are no duplicate values in rows, leaving only columns and sub-grids to be solved.

1.3 Fitness Evaluation

Duplicates in columns and sub-grids are added together to form the Fitness value. A higher value indicates a worse fitness, and a Fitness value of 0 indicates the solution to the Sudoku problem. Again, rows are kept in tact and are thus ensured to have no duplicates after the initial creation (no need to count them in the fitness function).

1.4 Selection Method

Tournament selection is used (the tournament size is set as a run time argument). Each tournament randomly selects elements from the population (as many as indicated in the tournament size), the candidate with the best fitness continues on to the next generation's population. The population size devided by the tournament size determines the amount of tournaments to be held *(tournamentCount = tournamentSize/populationSize)*.

1.5 Genetic Operators Used

1.5.1 Crossover

Two parents are randomly selected from the new population (after Tournament Selection). Offspring is created by randomly selecting which rows from each parent to use in the offspring. i.e.
ParentA Row1
ParentA Row2
ParentB Row3
ParentA Row4
ParentB Row5
etc.

1.5.2 Mutation

A percentage (provided as runtime argument) of the newly created population (created via Selection & Crossover) is mutated. The candidates to mutate are randomly selected.

1. **Row-Mutation**
   Random rows are selected and shuffled to create a new valid permutation of 1..9 (keeping the input values fixed).

2. **Smart-Row-Mutation**
   A random row is selected, and its columns scanned. If a value is found to be a duplicate in a column it is swapped with another duplicate in the same row.

1.5.3 Elitism

Provided as a run time argument, the best selection of the previous population can be forced into the next.

1.5.4 Other optimisations

Provided as a run time argument, Hidden and Naked Singles can be filled in before the GA is applied. This speeds up the time required to find the solution.